

Examensarbete
LITH-ITN-MT-EX--2002/19--SE

Visualising the Visual Behaviour of Vehicle Drivers

Björn Blissing

2002-10-21



TEKNISKA HÖGSKOLAN
LINKÖPINGS UNIVERSITET

LITH-ITN-MT-EX--2002/19--SE

Visualising the Visual Behaviour of Vehicle Drivers

Examensarbete utfört i Medieteknik
vid Linköpings Tekniska Högskola, Campus Norrköping

Björn Blissing

Handledare: Dennis Saluäär
Examinator: Björn Gudmundsson

Norrköping den 21 Oktober 2002



Avdelning, Institution
Division, Department

Medieteknik,
Institutionen för teknik och naturvetenskap
Media Technology,
Department of Science and Technology

Datum
Date

2002-10-21

Språk
Language

- Svenska/Swedish
 Engelska/English

Rapporttyp
Report category

- Licentiatavhandling
 Examensarbete
 C-uppsats
 D-uppsats
 Övrig rapport

ISBN

—

ISRN

LITH-ITN-MT-EX--2002/19--SE

Serietitel och serienummer ISSN

Title of series, numbering —

URL för elektronisk version

www.ep.liu.se/exjobb/itn/2002/mt/019/

Title Visualising the Visual Behaviour of Vehicle Drivers

Titel Visualisering av visuellt beteende hos fordonsförare

Författare Björn Blissling
Author

Sammanfattning

Abstract

Most traffic accidents are caused by human factors. The design of the driver environment has proven essential to facilitate safe driving. With the advent of new devices such as mobile telephones, GPS-navigation and similar systems the workload on the driver has been even more complicated. There is an obvious need for tools supporting objective evaluation of such systems, in order to design more effective and simpler driver environments.

At the moment video is the most used technique for capturing the drivers visual behaviour. But the analysis of these recordings is very time consuming and only give an estimate of where the visual attention is. An automated tool for analysing visual behaviour would minimize the post processing drastically and leave more time for understanding the data.

In this thesis the development of a tool for visualising where the driver's attention is while driving the vehicle. This includes methods for playing back data stored on a hard drive, but also methods for joining data from multiple different sources.

Nyckelord 3D Visualisation, Visual Attention, Visual Behaviour, OpenSceneGraph, Vehicle Drivers
Keywords

Abstract

Most traffic accidents are caused by human factors. The design of the driver environment has proven essential to facilitate safe driving. With the advent of new devices such as mobile telephones, GPS-navigation and similar systems the workload on the driver has been even more complicated. There is an obvious need for tools supporting objective evaluation of such systems, in order to design more effective and simpler driver environments.

At the moment video is the most used technique for capturing the drivers visual behaviour. But the analysis of these recordings is very time consuming and only give an estimate of where the visual attention is. An automated tool for analysing visual behaviour would minimize the post processing drastically and leave more time for understanding the data.

In this thesis the development of a tool for visualising where the driver's attention is while driving the vehicle. This includes methods for playing back data stored on a hard drive, but also methods for joining data from multiple different sources.

Sammanfattning

De flesta trafikolyckor orsakas av mänskliga faktorer. Designen av förarmiljön har visat sig mycket viktig för att underlätta säker körning. I och med introducerandet av tillbehör såsom mobiltelefoner, GPS-navigationssystem och liknande system har förarmiljön blivit mer och mer komplicerad. För att kunna designa effektiva, säkra och enkla gränssnitt så behövs det metoder för att objektivt utvärdera dessa.

För tillfället används oftast video för att fånga förarens visuella beteende. Men analysen av dessa inspelningar tar väldigt lång tid och ger inte särskilt noggranna resultat. Ett automatiserat verktyg för att analysera ögonbeteende skulle minimera efterarbetet drastiskt och ge mer tid för förståelse.

I detta arbete presenteras utvecklingen av ett verktyg gjort för att visualisera var föraren har sin koncentration under körning. Detta innefattar uppspelning av data loggad till hårddisk, men även metoder för att slå samman och spela upp data ifrån flera olika källor.

Preface

Thesis outline

This report first gives a brief introduction into methods for measuring visual demand, defining some concepts that are used throughout the thesis. Then data recording from other sources are explained focusing on test cars and simulators. After that the problems arising when joining two data files are explained and a possible solution to the problem is presented. Following these chapters the implementation of the visualisation is presented. In the end possible improvements to the implementation are discussed.

Acknowledgment

I would like to thank a number of people, which helped making this thesis possible:

- Dennis Saluäär, my supervisor at Volvo Technology Corporation.
- Peter Blomqvist, for helping me with all sorts of coding problems.
- Dr Björn Gudmundsson, my examiner at Linköping University.

I would also like to thank the whole Human System Integration Department at Volvo Technology Corporation for their great support.

Contents

Abstract	VII
Sammanfattning	VII
Preface and Acknowledgment	IX
1 Introduction	1
2 Measurement of Visual Behaviour	5
3 Car Data Sources	15
4 Combining Data	19
5 Implementation	23
6 Final Comments and Conclusions	33
References	35
A FaceLAB Performance Table	37
B FaceLAB Software Loop	39
C Scene Graph	41

List of Figures

1.1	Diagrams of visual behaviour.	3
2.1	A typical video setup.	8
2.2	Electro-oculography.	9
2.3	Scleral search coils.	10
2.4	FaceLAB 3D coordinate systems.	12
2.5	FaceLAB 2D coordinate systems.	13
3.1	The VTEC simulator.	16
4.1	Mending holes.	21
5.1	The developed Active X components.	26
5.2	The flowchart for the timer algorithm.	29
5.3	Data pipelines.	30
5.4	A screenshot from the developed program.	31
B.1	FaceLAB software loop.	40
C.1	The scene graph of the program.	42

Chapter 1

Introduction

Volvo Technology Corporation

Volvo Technology Corporation (VTEC) is an innovation company that, on contract basis, develops new technology, new products and business concepts within the transport and vehicle industry. The primary customers are the Volvo Group companies and Volvo Car Corporation but also some selected suppliers.

The R&D work involves a number of basic areas, e.g. transportation, telematics, internet applications, databases, ergonomics, electronics, combustion, mechanics, industrial hygiene and industrial processes, and using techniques such as systems engineering, multi-physical and chemical modelling, programming and simulation. Beside the R&D areas, VTEC offers specialist services in the areas of intellectual property protection, standardisation and information retrieval.

The department of Human System Integration works with optimising the interaction between the driver and the vehicle. The principal goal is to ensure a safe and efficient relationship between the user and the vehicle system. Human System Integration has a multidisciplinary staff of employees skilled in ergonomics and engineering sciences.

The research and development activities are within the areas of User impressions and satisfaction, Driving support, Driver awareness support, Interaction design and Driving simulation and Virtual Reality.

Background

It has been shown that inattention accounts for an estimated 25-56% of all accidents [17]. In order to design safe vehicles it is important to be able to evaluate in-vehicle systems to determine how distracting they are while driving.

There exist numerous studies that have established the relationship between eye movements and higher cognitive processes [5, 10]. These studies argue that the

eye movement reflects, to some degree, the cognitive state of the driver. Also, in several studies, eye movement are used as a direct measurement of driver attention and mental workload [5, 11].

As the number of in-vehicle system increases so does the visual demand on the driver. The design of these in-vehicle systems must therefore require a minimum of visual demand, e.g. good ergonomics. In order to define good design some objective test methods have to be used.

To be able to test the visual demand usually some kind of eye tracking has to be done. Earlier these method has been either very intrusive, inaccurate or very hard to use outside a laboratory environment. But now other methods based on computer vision are available. These methods are neither intrusive like some of the earlier methods nor do they require special lighting conditions, but have until now been very noisy outside a laboratory environment. The computer vision method used in capturing the data that are used in this thesis is uses a template matching technique. This method is robust and can be used outside a laboratory environment.

A study [16] has been made that statistically verified the computer vision eye tracking method with the ISO and SAE standards and work [7] has been done to fully automate the process described in ISO 15007-2 [1].

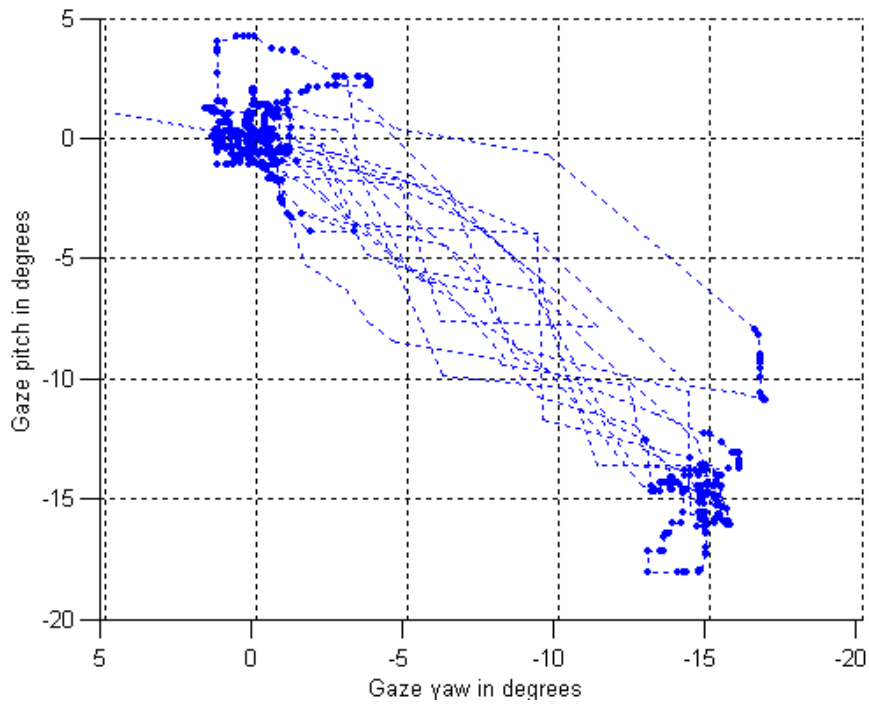
Motivation

The result from the tracking data is normally presented as diagrams or tables of data (See fig. 1.1). Such diagrams give good overall information of the test, but give no information related to transition patterns neither does it give any feedback on how the visual distraction affects the drivers behaviour.

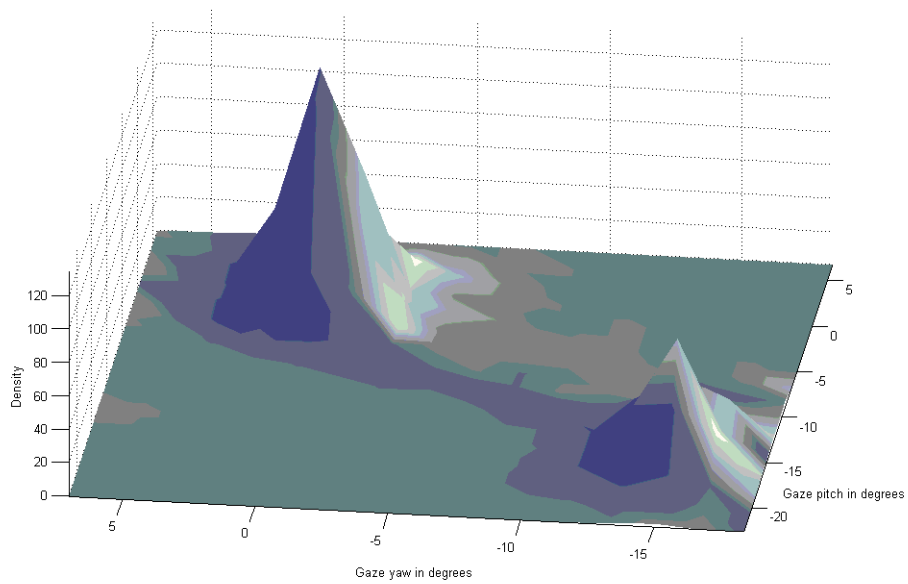
Performance data from the simulator or test car is often recorded i.e. speed, steering angle etc. The idea of this thesis is to use that data combined with the eye tracker data to achieve even more knowledge about the drivers' behaviour.

The goal of this thesis is to develop a piece of software that visualises this data as graphical elements in both 2D and 3D hence making the understanding of the data easier and better.

At Volvo Technology a prototype of a similar program has been developed, but since it was made using Performer on an IRIX workstation there was a request for a Windows version. The beta program also lacked a GUI and required some preprocessing of the data.



(a) Fixation and transition.



(b) Density plot.

Figure 1.1: Diagrams of visual behaviour.

Chapter 2

Measurement of Visual Behaviour

In order to define good design some objective test methods have to be used. It is also important that those tests uses common metrics to be able to compare one test to another.

Ocular motion

Ocular motion can be divided into different categories e.g. saccades, microsaccades, smooth pursuit, vergence, tremor, drift, etc. The understanding of all different types is beyond this thesis. In this work a generally accepted high-level description of ocular motion is used, dividing ocular motion into two fundamental categories: saccades and fixations.

There are different opinions on what defines fixations and saccades [4, 12]. In this thesis fixations are defined as pauses over informative regions. To be a valid fixation the pause has to be at least 150 ms at the specified region.

Saccades are the rapid movements (up to $700^\circ/\text{s}$) when the eye changes the point of view. During saccadic movement our brain does not perceive any information because the light moves too fast over the retina [4]. This means that saccadic movements of the eyes should be minimized while driving and all user interfaces should be designed so they require a minimum of change in the point of view.

Another type of fixation is smooth pursuit. This is when the eyes track a moving object such as an approaching road sign or a pedestrian on the sidewalk. These movements of the eyes are usually slow (within 80 to $160^\circ/\text{s}$).

Ocular measure

There exists standards [1, 6] that give definitions and metrics related to measurements of ocular motion of vehicle drivers. These measures are used when analyzing the data captured during tests of driver visual behaviour.

Basic ocular measures

The most basic ocular measures are:

- Saccade – The rapid movement changing the point of view
- Fixation – Alignment of the eyes so that the image of the fixated target falls on the fovea for a give time period.
- Average fixation duration – The average time the fixations lasts.
- Eye closure – Short eye closures indicate blinks whereas long closures could indicate drowsiness.

Glance-based measures

The glance-based measures are considered a higher-level description of eye movements. These measures reflect different properties such as time-sharing, workload and visual attention. Typical glance-based measures are:

- Glance duration – The time from which the direction of gaze moves towards a target to the time it moves away from it.
- Glance frequency – The number of glances to a target within a predefined sample time period, or during a predefined task, where each glance is separated by at least one glance to a different target.
- Total glance time – The total glance time associated with a target.
- Glance probability – The probability for a glance at a specified target.
- Dwell time – Total glance time minus the saccade initiating the glance.
- Link value probability – The probability of glance transition between two targets.
- Time off road-scene-ahead – The total time between two glances to the road scene ahead, which is separated by glances to non-road targets.

- Transition – A change in eye fixation location from one target location to another target location i.e. the saccade initiating a glance.
- Transition time – The duration from the end of one fixation to the start of new fixation on another target.
- Total task time – Total time of a task, defined as the time from the first glance beginning to last glance termination during the task.

Measurement of Visual Behaviour

There exists a number of different methods to measure the visual behaviour of the driver:

Notes

The simplest method to record the driver's attention is for the test leader to manually make notes of where the driver is looking. This can be an exhausting task and it is hard to get accurate gaze direction. The accuracy highly depends on the ability of the test leader to make a good estimation of the gaze direction. Many factors like change in speed and lane offset are omitted which may leave out data that could be important for making a good analysis.

The advantage of this method is that it does not require any special hardware to be purchased and mounted in the test car.

PDT

The Peripheral Detection Task (PDT) is a method for measuring the amount of mental workload and visual distraction in road vehicles. It is a task where the drivers must respond to targets presented in their peripheral view. As drivers become distracted they respond slower and miss more of the PDT targets.

The PDT targets usually are a number of light emitting diodes placed on the dashboard at approximate 11 to 23 degrees to the left of the drivers forward view. The LEDs light up with a random variation of 3-6 seconds. The drivers then have to respond within 2 seconds otherwise it is considered a miss. The response is usually given by pressing a button that is placed on the test subject's index finger.

The method records both reaction time and hit rate. It has been shown in numerous tests [3, 9, 15] that both reaction time and hit rate drops when the driver is distracted.

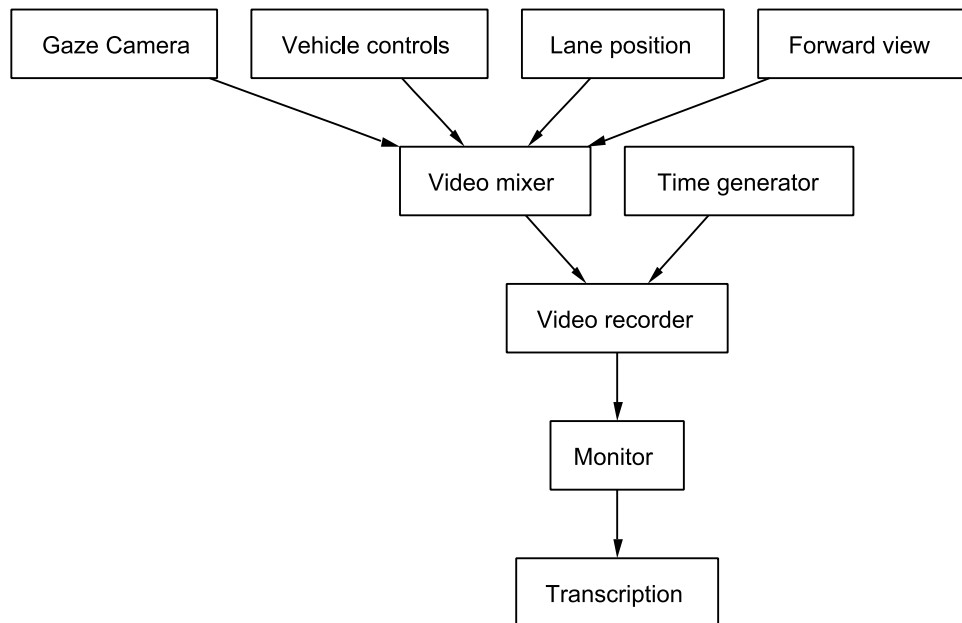


Figure 2.1: A typical video setup.

The PDT method is a statistical method where mean reaction time and hit rates are considered. This gives good information about the overall mental workload when performing tasks, but it does not give any information on ocular measures.

Video

The method (described in [1]) is based on four video cameras (preferably small, one chip CMOS cameras). These cameras can be set up differently but the most common is to have one camera viewing the driver, one viewing the road, one viewing the vehicle controls and one camera monitoring the lane position (See Fig. 2.1). These images are multiplexed to a single frame and recorded together with a time code on a high quality video recorder.

After the test the video has to be analyzed where interesting parts of the data is transcribed into notes. This requires that the person transcribing the video can manually determine the drivers gaze direction, which can be difficult. The work of transcribing the video can be very time consuming (One hour video takes approximate one day to transcribe.).



Figure 2.2: Electro-oculography.

Factors like change in speed and lane offset are normally omitted when using this method, but an external logging system can be used to capture the data although no combined playback of video and logged data can be performed on existing systems.

EOG

EOG (electro-oculography) is a method where the potentials, induced by the small muscles which control the eyes, are measured using small electrodes (See fig. 2.2).

The method is not very accurate and only measures the rotation of the eyes. If the position and rotation of the head is desired additional headtracking equipment has to be used.

Scleral search coils

This method uses small coils attached (usually with vacuum) to the eyes. Their positions independent of head movements are measured by placing the subject between two larger coils, which induce voltage in the small coils (See fig. 2.3).

This method could be painful for the test subject and usually the eye has to be anaesthetised externally to be able to attach the search coils. The fact that the method could be painful to the test subject makes it less reliable, since the test person could change his visual behaviour because certain movements of the eye cause pain.

Since the method is based on measuring small induced voltage the test has to be performed in a controlled environment to be accurate.

Corneal reflection

In this method the test subject is exposed to infrared light. A small camera measures the amount of the infrared light that is reflected in the cornea. The amount of reflection depends on the angle of the eye.

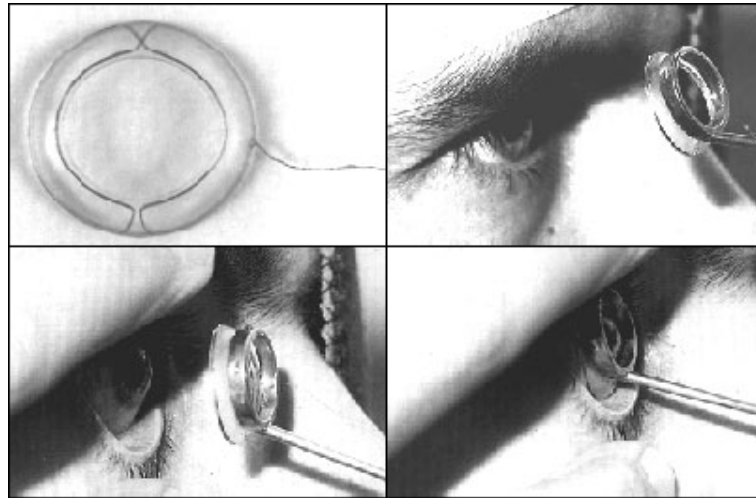


Figure 2.3: Scleral search coils.

The method is very accurate, but only measures the rotation of the eyes. If the position and rotation of the head is desired additional head tracking equipment has to be used. It is also very sensitive to changes in light, making it hard to use outside a laboratory environment.

FaceLAB

FaceLAB [14] is a product for non-intrusive measuring of head pose and gaze direction. It is developed by Seeing Machines with sponsorship from Volvo Technology. Seeing Machines is a spin-off company from the Research School of Information Sciences and Engineering at the Australian National University. This is the method used to capture all the head position and gaze data used in this thesis.

The method [8] they have developed is based on two monochromatic video cameras, which give a stereo pair image. These cameras are mounted either on a steel frame, called stereohead, or directly into the car. The two cameras register the driver's face and track a number of features in the face. Features especially suitable for tracking are corners of the eyes and mouth. (Markers on the face can be used as they are even better features to track than ordinary facial features.) Since the system uses two cameras, 3D coordinates for each feature can be calculated.

For tracking of gaze direction the system uses a model of the eyes where the eyes are considered as perfect spheres. The gaze direction is determined based on both the pose of the head and the position of the irises of the eyes. By calculating the position of the iris on the eye, the horizontal and vertical rotation can be estimated i.e. the gaze direction.

There are four eyes in total in the stereo image pair; therefore four gaze directions are detected independently. However each measurement is not sufficiently accurate, mainly due to the resolution of the input images from the cameras (The width of the eye is only about 30 pixels). Therefore it is hard to determine the gaze point in a 3D scene by calculating the intersection of the detected gaze lines. Instead those four vectors are averaged to a single normalized gaze vector in order to reduce the effect of noise. (See Appendix B.1 for flowchart of the FaceLAB algorithm.)

The method used in FaceLAB is robust to occlusions, deformations and fluctuations of lighting. Other advantages are that it works in real-time and it is accurate. (See Appendix A)

FaceLAB Coordinate Systems

FaceLAB makes use of five coordinate systems. It's world coordinate system has its origin between the two cameras. All position output is in world coordinates (See Fig. 2.4(a)). All rotations are outputted as Euler angles about un-rotated axes, with the order (pitch, heading, roll).

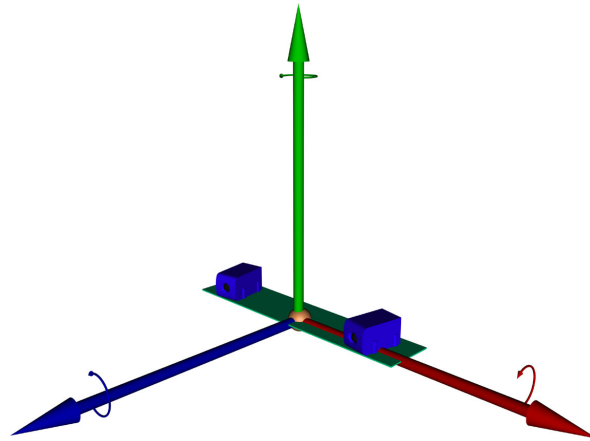
The next system is the stereohead coordinate system (See Fig. 2.4(b)). This coordinate system is coincident with the world coordinate system, but is attached to the stereohead, so it tilts with the stereohead.

The head coordinate system (See Fig. 2.4(c)) has the head as origin, with the x-axis aligned with the center of the eyes.

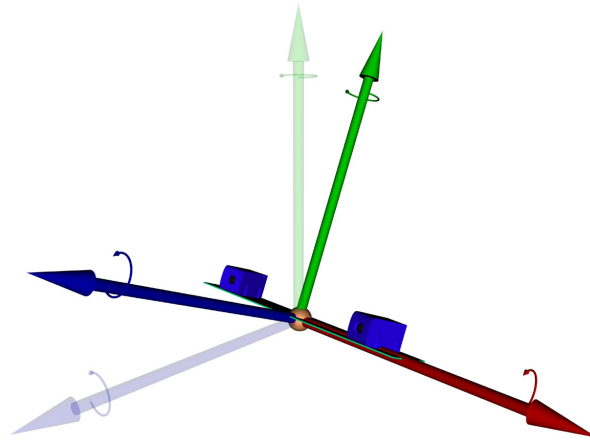
The two last coordinate systems are the screen coordinate system and pixel coordinate system. These are 2D systems used to measure where the person is looking at the screen (or environment).

The first 3 of these coordinate systems are right handed coordinate systems. Since rotation of the eyes is given only in 2D (it is not possible to roll the eyes) it is not possible to specify this as a left or right handed system. FaceLAB outputs the rotation of the eyes as pitch and heading independent of the rotation of the head.

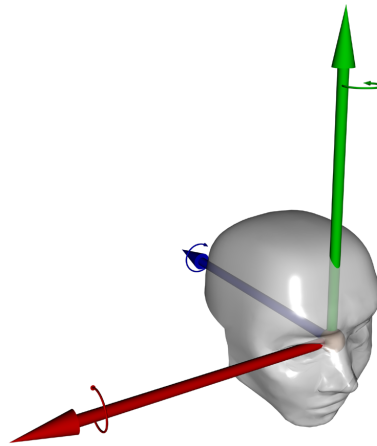
FaceLAB can also output gaze data in a screen coordinate system (See Fig. 2.5(a)) and a pixel coordinate system (See Fig. 2.5(b)). The screen coordinate system is chosen so it defines positive x as right and positive y as up with the origin in the middle of the screen, while the pixel coordinate system defines the origin in the top left corner with x as right and y down.



(a) FaceLAB world coordinate system.

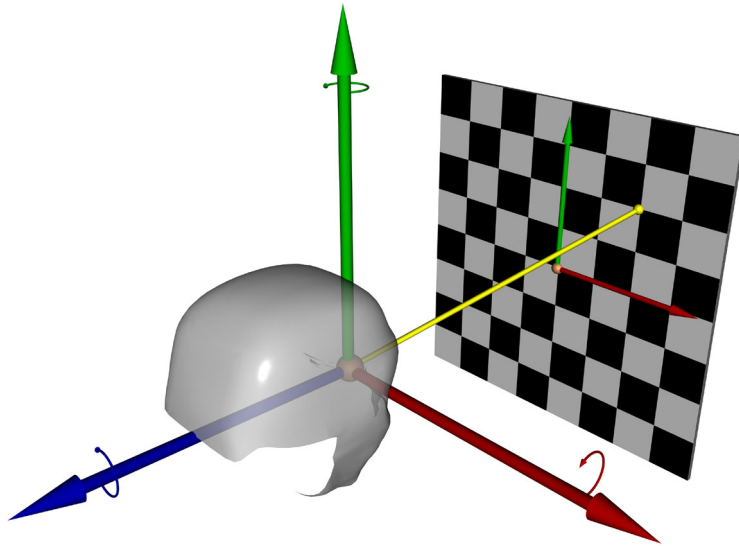


(b) FaceLAB stereohead coordinate system.

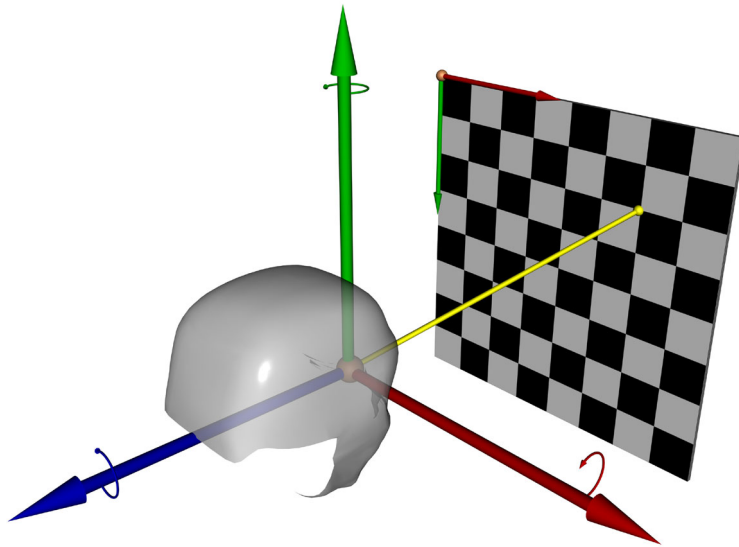


(c) FaceLAB head coordinate system.

Figure 2.4: FaceLAB 3D coordinate systems.



(a) FaceLAB screen coordinate system.



(b) FaceLAB pixel coordinate system.

Figure 2.5: FaceLAB 2D coordinate systems.

Problems with FaceLAB

Although FaceLAB is more robust than its competitors it still has some problems. The biggest problem is error in tracking accuracy when the cameras are exposed to quick changes in lighting conditions and even more when lens flares, due to direct sun light into the cameras, appears. Other problems such as undetected eye closures could also result in occasional glances towards objects that are unlikely to be focused e.g. test subjects knees, inner ceiling of the vehicle.

Signal processing of FaceLAB data

FaceLAB provides its own methods for identifying fixations and saccades, but it is an ad-hoc implementation and does not concur with any standards.

Work [7] has been done to implement a version of the fixation and saccade detection that follows the ISO standard. This work also includes filtering the data since the data from FaceLAB can be noisy and contain illegal fixations i.e. detected fixations that do not last at least 150 ms. At the same time as the filtering, the gaze data can be segmented into information for glance-based measures. That means dividing the fixations into clusters of different classes depending on glance direction e.g. on road, off road, center stack, instrument cluster etc.

Chapter 3

Car Data Sources

Besides the head- and eye tracker data other data sources could be used to get even more information of a driving test. The tests are performed either on a test car, test bench or in the VTEC simulator. Typical logging signals captured are speed, gear, steering angles, brake- and thrust power.

Test Car

The test car is usually a standard car with a computer running xPC Target connected to the CAN bus. The CAN bus contains most of the signals in the car. From the xPC computer signals can be sent over UDP to a logging computer.

CAN

Controller Area Network (CAN) is a serial data communications bus for real-time applications and specified in ISO 11898. It was originally developed by Bosch for use in cars, but is now being used in many other industrial automation and control applications. Volvo has been using CAN since the release of Volvo S80.

xPC Target

xPC Target is a system that enables a user to run an application created in Simulink or Stateflow on a real-time kernel. It is created by MathWorks [20].



Figure 3.1: The VTEC simulator.

Lane tracking

Lane tracking is supported with a product from AssistWare Technology called SafeTRAC [2]. This uses a small camera to register the road. From the registered picture data, such as lane offset and lane width, are derived through the use of image processing. The logged data is stored on a PCMCIA card drive connected to the unit.

The VTEC driving simulator

The VTEC simulator is a fixed-based simulator, which consists of a complete Volvo S80 mock-up and a cylindrical screen with a height of 3 meters and a radius of 3.5 meters (See fig. 3.1). The visual field-of-view presented for the driver/subject was 135 degrees. The simulations are run on a Silicon Graphics [26] Onyx2 with 16 CPUs each running at 250 MHz and three graphics channels of type Infinite Reality 2. The graphics is rendered in real-time at a maximum of 60 frames per second using OpenGL Performer. The road environments are modelled using MultiGen Creator [21]. Data can be sent from the simulator to a logging computer by UDP.

Test bench

When experimental displays or gauges are to be tested, physical models are usually not built. Instead a test bench is used. This is usually a simple mock-up where TFT screens display the gauges. The flexibility of such a mock-up is one of the advantages; new displays could easily be moved, changed, replaced or removed. Another advantage is that a test bench is very cost effective compared to physical models. The test bench can perform logging either by storing data directly on the test computer or sending the data over UDP as the simulator.

Chapter 4

Combining Data

The main idea with this thesis is to visualise head and eye tracker data together with performance data from the simulator or a test car. It is obvious that there is need for some form of merge of the data from the different sources.

The datasets

FaceLAB outputs data in either binary form or as MatLab files. The simulator outputs its data as tab separated textfiles. Both FaceLAB and the simulator have the possibility to output its data as a network stream over UDP.

Since the FaceLAB and the Simulator are two independent systems the data output is not completely homogenous. The only data they have in common is system time.

Interpolation

Since the data from the simulator and the data from FaceLAB are sampled independently there can be a difference in the rate of sampling and/or exact time of the collection of the sample. This is mainly due to different latencies in the systems.

To combine the data some interpolation must be done. Two different solutions have been considered. Either interpolating one of the two data files to the sample rate of the other file or to interpolate both files to a new common sample rate. Since both data streams are sampled at a non-uniform rate it was judged best to interpolate both of them into a new common sample rate.

Nearest neighbour

The simplest interpolation is the method of nearest neighbour. This method does exactly as it says - it takes the value in the point of the nearest neighbouring point. Nearest neighbour is suitable to interpolate Boolean data.

Linear

Linear interpolation takes one point above and one point below the searched value. The searched value is then calculated as a linear approximation from these two values.

Higher order interpolation

Higher order interpolation types such as spline, cubic or sinc interpolation could be also used. But the computational time and added implementation complexity needed for these does not improve the quality of the interpolation enough to motivate their use in this case.

Mending holes

If recording of the simulation data, for some reason, pauses it does not mean that FaceLAB automatically pauses as well and vice versa. This can give “holes” (See fig. 4.1(a)) in one of the datasets, which can result in problems when joining the two datasets.

Four solutions were considered to mend the holes. All methods start with making sure that both datasets start and end at the same time. If one of the datasets starts before the other, the preceding data is removed. The same thing is made in the end of the datasets – the longer dataset gets trimmed to give the datasets a common start- and end time.

The simplest method considered was to omit data that did not have any data at the corresponding time in the other dataset (See fig. 4.1(b)). One problem with this method is that it could give big skips in the time when playing back the data.

To provide a continuous time in the joined dataset some mending of holes or interpolation have to be applied. The second method considered involved interpolation between the last cell before the hole and the next cell after the hole (See fig. 4.1(c)). But this is not appropriate in this case. It could easily give the illusion that the dataset has more information than it actually has.

So instead a third method was suggested. The last data cell before the hole is copied into empty cells in the hole (See fig. 4.1(d)). This would give the illusion

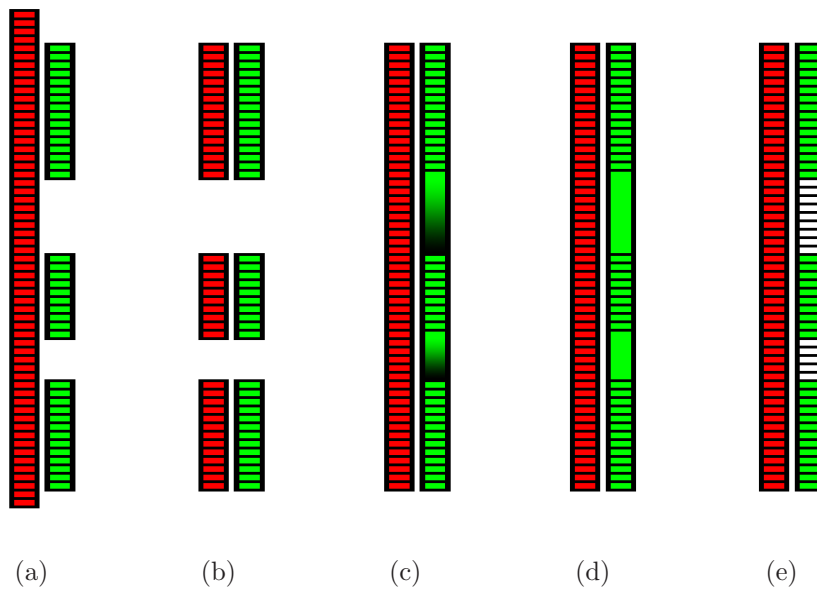


Figure 4.1: Mending holes.

that the recording of the data freezes but can still give the illusion that there is data that in fact do not exist.

The fourth method just fills the holes with zeros (See fig. 4.1(e)). This rules out that any new data is applied to the dataset. This could give a somewhat discontinuous data, but is the method most suitable for this application since only data that has been recorded is presented.

Chapter 5

Implementation

The demand from VTEC was that the final program ran on a Windows PC computer. Since most of the visualisation models at VTEC is in OpenFlight format, there was a desire for the program to read these models without any converting.

The development platform was a PC running at 1.3 GHz with a GeForce3 graphics card. The development was made using Visual C++ 6.0.

Graphics

Before starting the implementation several graphic packages were considered. Among those considered were:

- World Toolkit from Sense8[25]. Quickly rejected because of very high licensing cost (Development license \$10000).
- Gizmo3D from ToolTech Software[28] (Swedish). Looks promising but was not chosen because of licensing cost. (Development license \$1000 and runtime licence \$100 per machine)
- OpenSG[22] (NOT the same as OpenSceneGraph) was not chosen because it did not have support for reading OpenFlight models. Neither was Visual C++ 6.0 supported.
- PLIB[24] (Steves Portable Game Library) supports both Visual C++ 6.0 and loading of OpenFlight models. But the view frustum setting was hard to set to get a correct perspective projection. There also existed some problems with material attributes when loading OpenFlight models.
- OpenSceneGraph [23] a free and open source project. Supports many operating systems, object- and image formats. Although still in beta state.

Finally OpenSceneGraph was chosen for the project. Mainly because of the good support of the OpenFlight format, but also because VTEC expressed an interest in seeing what capabilities were available in OpenSceneGraph. Furthermore VTEC wanted to obtain knowledge about how difficult it would be to use in development of new software.

OpenSceneGraph

OpenSceneGraph is an open source initiative to try to make a platform independent scene graph package. Currently it supports IRIX, Linux, FreeBSD, Solaris and Windows. Some work has been done on HP-UNIX and Mac OS X.

Among its features is both quad buffered stereo and anaglyphic stereo. More interesting features are multi-texturing, cube mapping, occlusion culling, small feature culling, anisotropic texture filtering and particles.

OSG has loaders which supports 3D studio object files, Lightwave objects, SGI Performer, OpenDX, Terrapage and Open Flight. It also supports many image formats: bmp, gif, jpeg, SGI rgb, pic, png, tiff and tga.

Converting coordinates

The coordinate system in OpenSceneGraph is a right hand system, just like FaceLAB. The difference is that FaceLAB defines it's coordinate system as X - sideways, Y - up and Z - out. Whilst OpenSceneGraph defines it's as X - sideways, Y - in and Z - up i.e. the difference is a rotation of 90° about the X-axis. This difference can easily be overcome in the implementation by setting the following conversions:

FaceLAB		OpenSceneGraph
X	\iff	X
Y	\iff	Z
Z	\iff	$-Y$

Using Euler angles

In 3D space rotations consists of three components heading (sometimes called yaw), pitch and roll. When using the euler angles the order of successive rotations is significant. A simple example to illustrate this:

- Rotate 90° about X axis
- Rotate 90° about Y axis
- Rotate -90° about X axis

This would give a 90° rotation about Z axis even though no rotation about the Z axis were specified, whereas:

- Rotate 90° about X axis
- Rotate -90° about X axis
- Rotate 90° about Y axis

This would give a 90° rotation about Y axis (The first two lines cancel out each other). So clearly the order of the rotation matters a lot.

The rotation output from FaceLAB uses rotations about the three axes in the order (X, Y, Z) i.e. $(pitch, heading, roll)$. Which in the implementation using OpenSceneGraph gives the order $(X, Z, -Y)$. But since OpenSceneGraph defines X - sideways, Y - in and Z - up this is still equivalent with the order $(pitch, heading, roll)$.

Tools

In the implementation Microsoft Visual C++ 6.0 has been used. One problem with this is that the STL version, which comes with VisualC++6.0, is not implemented according to the standard. This is a problem for OSG, since it makes proper use of Standard C++. The solution chosen to solve this problem is to use the STL library from STLport Consulting [27] instead of Microsofts STL implementation.

When implementing the program a choice had to be made between a pure dialog-based application or to use the Document-View architecture. Since the program should load files (documents) and the visualise (view) them, it seemed like a sensible choice to use the Document-View architecture.

The installer program was created using a freeware program called InnoSetup 3.0.2[19]. This is an install script compiler. For easier creation of these scripts a GUI script editor called ISTool [18] was used.

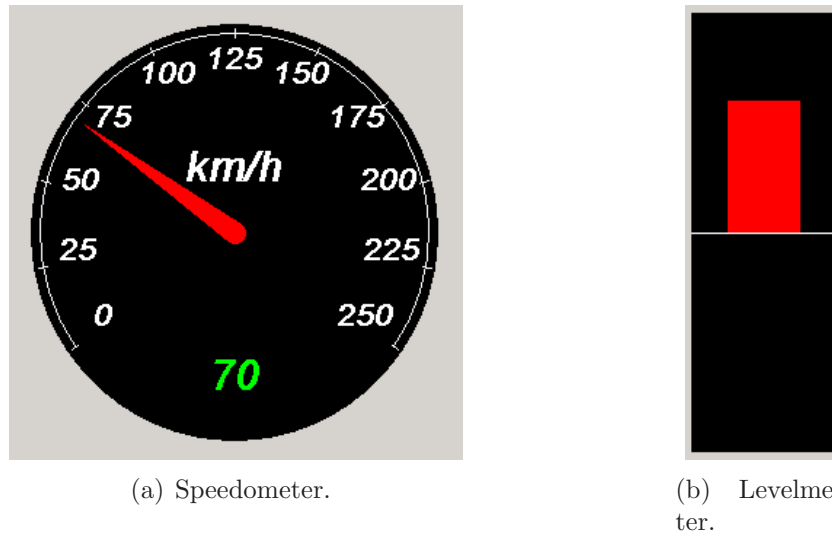


Figure 5.1: The developed Active X components.

User Interface

Even though OSG supplies a small set of its own GUI components the choice for the GUI was to make use of MFC components. Mainly because the users of the system quickly should feel acquainted with the standard windows GUI. But this choice also gave the possibility of using the GUI editor inside Visual C++.

ActiveX Components

Some data are inappropriate to visualise in 3D. Ordinary 2D meters are sometimes more accepted i.e. speedometers and level meters. Therefore these two meter-types were developed as Active X components and used in the program. (See fig. 5.1)

Visualization

The program developed can show three different 3D views. All views contain two elements; the road and the vehicle interior. The road is a small road piece model (about 20 m) that repeats infinitely whilst driving down it.

The first view, which is the explore view, contains both the road and vehicle interior but also a head model. This view lets the user zoom, pan and rotate around in 3D space. The head is shown with a normal vector pointing out in front of it. The gaze vector is visualised as a line emerging from a point between the eyes. An object-line intersection is calculated between the gaze vector and

vehicle interior and this point is highlighted. This dramatically helps the user to see where the driver's attention is. The gaze line intersection test is done in real time. This was one of the calculations in the prototype that required the extensive preprocessing.

The second view puts the user inside the driver's head. No panning, zooming or rotation is possible in this view. Instead this view automatically pans, zooms and rotates with the driver's head i.e. the recorded data from FaceLAB.

The third view gives the view from the driver's eyes. Changes in heading and pitch follow the movements of the driver's eyes. Other movements such as panning, zooming and rolling follow the movements of the head. This view can be very jerky because of noise from FaceLAB, but also because our brain normally filters our own microsaccades out. Seeing the world from someone else's eyes is not always a pleasant experience.

Other data such as speed, acceleration, steering angles, thrust- and brake pressure are visualised with the Active X components mentioned before. Higher order glance based measures i.e. the segmented clusters of data, can be visualized using small LED like displays.

The program is able to playback, pause, rewind, fast-forward and jump anywhere in the loaded data set. This is supported through buttons with ordinary video recorder symbols and a time slider bar.

All models in the program is exchangeable, provided that they are supplied in a format that OpenSceneGraph is capable of reading. The static displacement of the models can also be changed. This means that a car interior quickly can be replaced with the interior of a lorry, compensating for the added height of the driver cabin in the lorry. The background colour can be interactively changed and the fog effect can be turned on and off. All these settings are stored as register keys in the Windows register.

Timing of playback

When visualising the data an algorithm is necessary to ensure timing of the playback. The algorithm should ensure that the playback rate stays coherent with the recording rate. This means that the algorithm must take the rendering time into account. Although skipping of frames is not allowed since we do not want to lose any data. This could give a slightly slower playback of the data if the recording frequency exceeds the rendering frequency. But it is still preferable to losing data in the playback.

The program uses a error diffusion like algorithm for keeping the playback rate coherent as possible with the recording rate, but ensuring that each data cell get displayed at least once. The algorithm (See fig 5.2) starts at the first cell in the data. It then checks the timestamp of the next cell and calculates how

long the current cell should be displayed before advancing to the next cell. The accumulated rendertime is set to zero and the rendering of the frame is done. The time of the render is added to the accumulated rendertime. If the accumulated rendertime is less than the time the current cell should be shown no advance is made, but if the rendertime is larger or equal it advances one step in the dataset.

Before rendering the new frame we subtract the accumulated rendertime with the time the cell has should be shown. This is done to ensure that the data playback does not lag behind more than necessary.

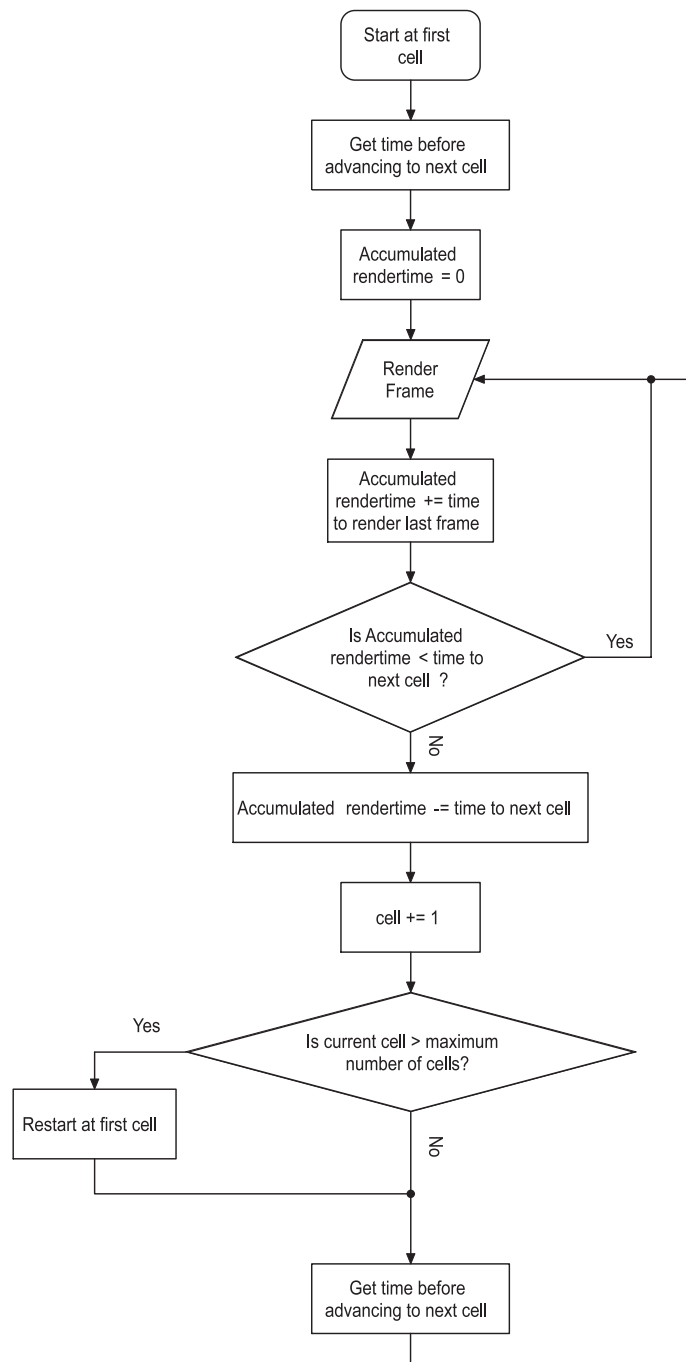


Figure 5.2: The flowchart for the timer algorithm.

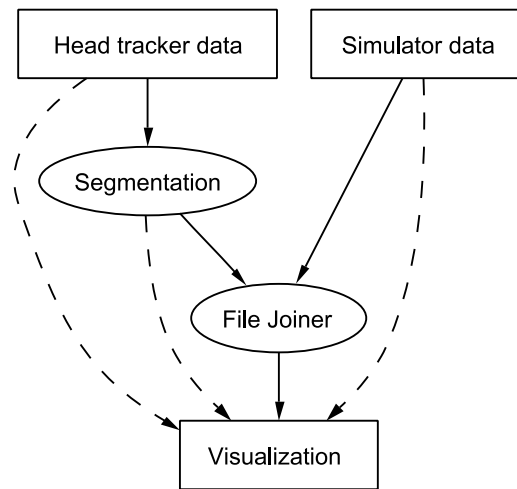


Figure 5.3: Data pipelines.

Data input

The program reads the data from tab- or space separated textfiles. The data can have its source either from FaceLAB or the simulator. But the program will be able to give more information when both sources are combined, which was the main idea when developing the program. (See fig. 5.3 for a conceptual drawing of the data pipeline. The dotted lines show possible data pipelines while the drawn line represents the preferred data pipeline.)



Figure 5.4: A screenshot from the developed program.

Data output

So far the only output from the program is the renderings to the screen (See fig. 5.4).

Chapter 6

Final Comments and Conclusions

Conclusions

The use of OpenSceneGraph proved to be a successful choice. Although still in beta state it proved to be stable and worked smoothly together with Windows.

One regret concerning the implementation was the choice of using the Document-View architecture. This proved to give no advantages at all and only made programming more complicated. Especially the timing routines became very hard to overview, since this algorithm had to be divided into two classes.

The developed program has received good reviews of the employees at VTEC. Especially the feature to view the world from the test subjects eyes has been very appreciated.

Future work and recommendations

Although the program developed could be used in it's present state some improvements would be good to implement:

- Enable the plotting of signal diagrams would give information regarding the history of the signals.
- Network playback. Instead of playing back recorded data, the program could be used to show the present states during a simulation by visualizing the current data received over an UDP stream.
- Binary file readers. This would remove the need to convert the logged files into tab separated text files.
- Enable saving of screenshots and videos from the program.

- A bigger project would be to replace the infinite straight road with a more realistic one. One solution could be to use the curvature values from the lane tracker to dynamically create a road, that looked more like the road where the actual test took place.

References

- [1] ISO 15007-2. Road vehicles - measurement of driver visual behaviour with respect to transport information and control systems, 2001.
- [2] AssistWare Technology. *SafeTRAC owner's manual*, 1999.
- [3] P.C. Burns, E. Knabe, and M. Tevell. Driver behavioural adaptation to collision warning and avoidance information. In *IEA/HFES, International Ergonomics Association*, San Diego, August 2000.
- [4] R.H.S. Carpenter. *Movements of the Eyes*. Pion, London, 2nd edition, 1998.
- [5] J.L. Harbluk and Y.I. Noy. The impact of cognitive distraction on driver visual behaviour and vehicle control. Technical report, Ergonomics Division, Road Safety Directorate and Motor Vehicle Regulation Directorate, Minister of Transport, Canada, 2002.
- [6] SAE J-2396. Definition and experimental measures related to the specification of driver visual behaviour using video based techniques, 1999.
- [7] P. Larsson. Automatic visual behavior analysis. Master's thesis ISRN: LiTH-ISY-EX-3350-2002, Department of Electrical Engineering, Linköpings Universitet, Sweden, 2002.
- [8] Y. Matsumoto and A. Zelinsky. An algorithm for realtime stereo vision implementation of head pose and gaze direction measurement. In *Proceedings of IEEE Fourth International Conference on Face and Gesture Recognition*, pages 499–505, Grenoble, France, March 2000.
- [9] S. Olsson. Measuring driver visual distraction with a peripheral detection task. Master's thesis ISRN: LIU-KOGVET-D-0031-SE, Department of Behavioural sciences, Linköpings Universitet, Sweden, 2000.
- [10] M.N. Recarte and L.M. Nunes. Effects of verbal and spatial imagery tasks on eye fixations while driving. *Journal of Experimental Psychology: Applied*, 6(1):31–42, 2000.

-
- [11] T. Ross. Safety first. In *Transport Technology Ergonomics Center*, Traffic Technology international, Loughborough University, UK, Feb/Mar 2002.
- [12] D. Salvucci and J. Goldberg. Identifying fixations and saccades in eyetracking protocols. In *Proceedings of Eye Tracking Research & Applications Symposium*, Palm Beach Gardens, FL, USA, 2000.
- [13] Seeing Machines. *FaceLAB 1.0 - Frequently Asked Questions*, 2001.
- [14] Seeing Machines. *User manual for FaceLAB v1.1*, 2002.
- [15] W. Van Winsum, M. Martens, and L. Herland. The effects of speech versus tactile driver support messages on workload driver behaviour and user acceptance. TNO-report TM-00-C003, Soesterberg, The Netherlands, 1999.
- [16] T. Victor, O Blomberg, and A. Zelinsky. Automating driver visual behaviour measurement. 2002.
- [17] J.S. Wang, R.R. Knippling, and M.J. Goodman. The role of driver inattention in crashes: New statistics from the 1995 crashworthiness data system. In *Proceedings of the 40th Annual Meeting of the Association of Automotive Medicine*, Vancouver, Canada, October 1996.
- [18] <http://www.bhenden.org/istool/>.
- [19] <http://www.jrsoftware.org/>.
- [20] <http://www.mathworks.com/>.
- [21] <http://www.multigen.com/>.
- [22] <http://www.opensg.org/>.
- [23] <http://www.openscenegraph.org/>.
- [24] <http://plib.sourceforge.net/>.
- [25] <http://www.sense8.com/>.
- [26] <http://www.sgi.com/>.
- [27] <http://www.stlport.org/>.
- [28] <http://www.tooltech-software.com/>.

Appendix A

FaceLAB Performance Table

Head tracking volume (height, width, depth)	0.2 m, 0.3 m, 0.5 m
Head position accuracy (x,y,z)	1 mm, 1 mm, 2 mm †
Head rotation accuracy (heading, pitch, roll)	2°, 2°, 1° rms ‡
Head rotation tolerance (heading, pitch, roll)	90°, 30°, 30° ††
Gaze direction accuracy	2° rms ‡‡
Working frequency	60 Hz
Temporal latency	~33 ms

† Best case when head is motionless

‡ Best case (worst case estimated to be about 5°)

†† Maximum rotation before tracking is lost. (FaceLAB can recover tracking when lost)

‡‡ Worst case 15°

Reference: [13]

Appendix B

FaceLAB Software Loop

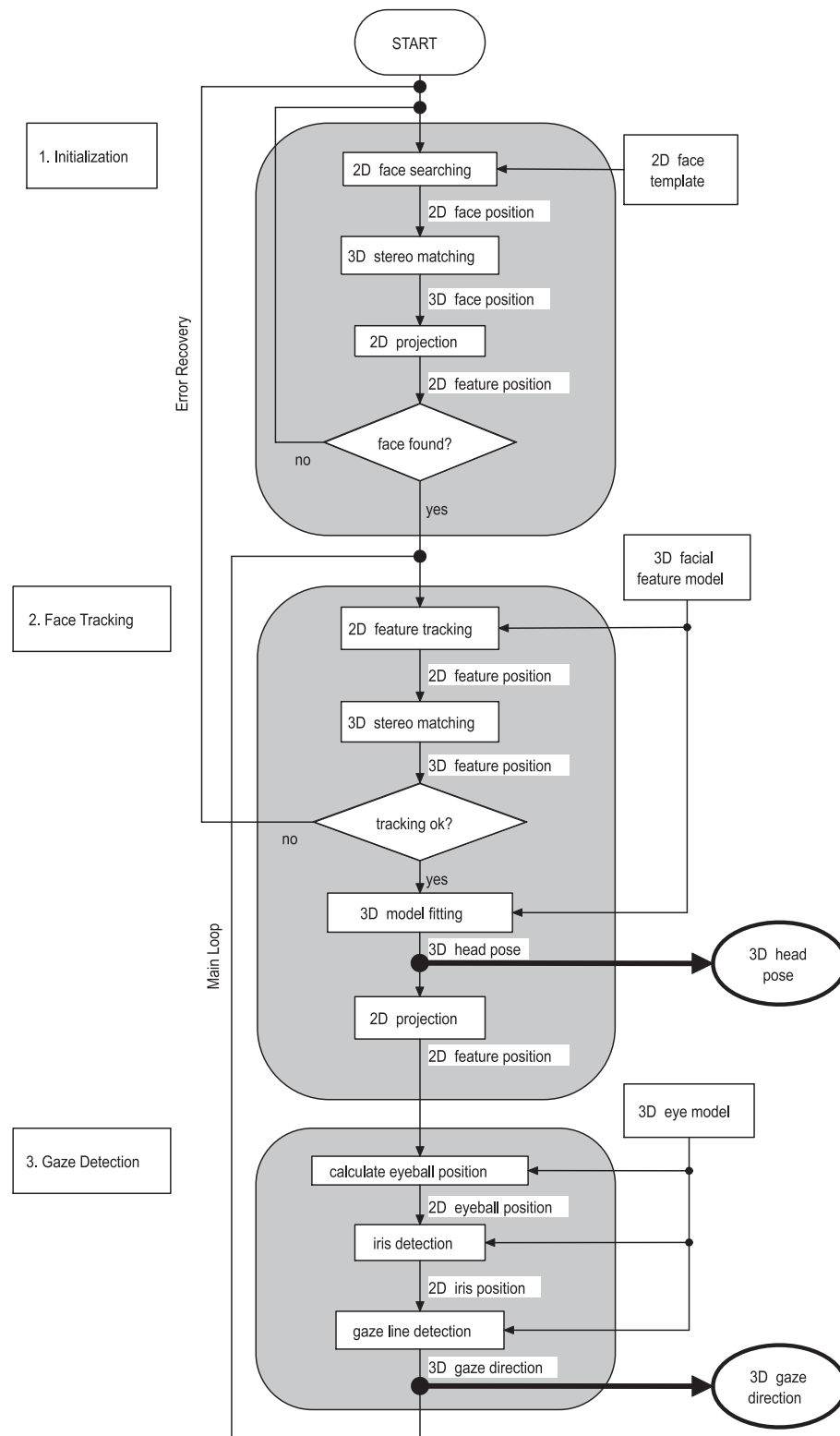


Figure B.1: FaceLAB software loop.

Appendix C

Scene Graph

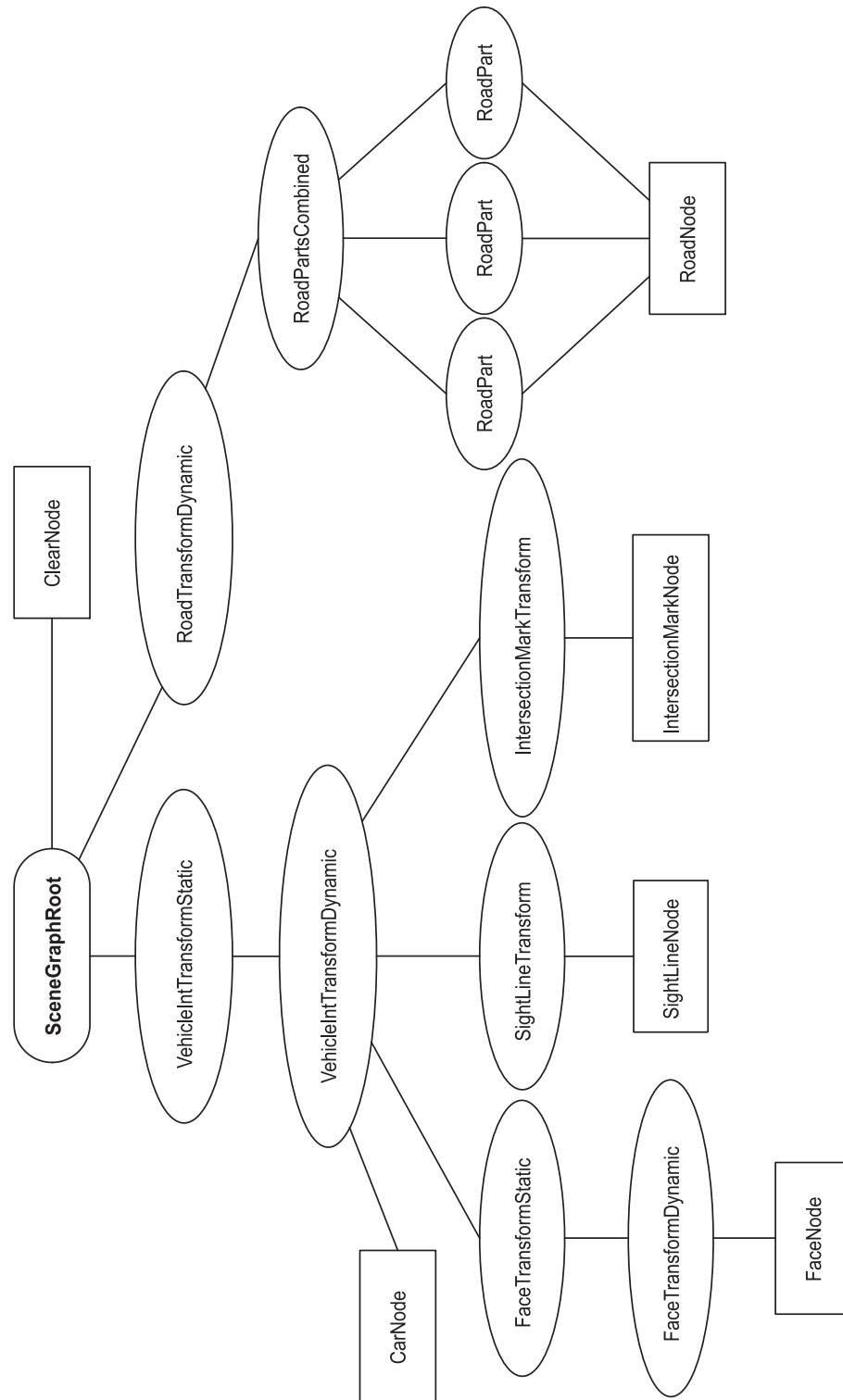


Figure C.1: The scene graph of the program.